Adaptive Finite Element Methods

Peter Binev, Wolfgang Dahmen, Ronald DeVore

University of South Carolina / RWTH Aachen

The Problem

$$-\Delta u = f$$

$$u = 0 \text{ on } \partial\Omega$$

Ω is a polygonal domain in $I\!R^2$

1. Standard Finite Element Methods
2. Adaptive Finite Element Methods

IS THERE ANY ADVANTAGE TO ADAPTIVE METHODS

Little Known

- 1998–2000: Convergence of certain AFEM

Little Known

- 1998–2000: Convergence of certain AFEM
- I know of no AFEM ($d \geq 2$) which is proven to outperform Standard Finite Element Methods for problems in $\mathbb{R}^d$, $d \geq 2$

Goal

- More ambitious

Goal

- More ambitious
- Give the ‘mother’of all adaptive algorithms

Galerkin Solutions

- P a triangular partition of Ω

Galerkin Solutions

- P a triangular partition of Ω
- $\mathcal{S}_P$ space of piecewise linear functions subordinate to P vanishing on $\partial\Omega$

Galerkin Solutions

- P a triangular partition of Ω
- $\mathcal{S}_P$ space of piecewise linear functions subordinate to P vanishing on $\partial\Omega$
- u_P Galerkin solution

Galerkin Solutions

- P a triangular partition of Ω
- $\mathcal{S}_P$ space of piecewise linear functions subordinate to P vanishing on $\partial\Omega$
- u_P Galerkin solution
- $\|u - u_P\| = \inf_{S \in \mathcal{S}_P} \|u - S\|$

Galerkin Solutions

- P a triangular partition of Ω
- $\mathcal{S}_P$ space of piecewise linear functions subordinate to P vanishing on $\partial\Omega$
- u_P Galerkin solution
- $|||u - u_P||| = \inf_{S \in \mathcal{S}_P} |||u - S|||$
- $||| \cdot |||$ energy norm (H^1-norm)

Typical Adaptive Algorithm: $P_k \longrightarrow P_{k+1}$

1. Compute u_{P_k}
2. Compute error indicators $e(\Delta)$, $\Delta \in P_k$. $e(\Delta)$ has two terms: (i) jumps in u_{P_k} accross edges of Δ, (ii) approximation of f by constants on Δ
3. Use local error indicators to mark cells $\mathcal{M}_k$ in P_k: bulk chasing.
4. Refine cells in $\mathcal{M}_k$ to obtain P'_k
5. Remove hanging nodes: further markings $\mathcal{M}'_k$
6. Refine cells in $\mathcal{M}'_k$: $P'_k \longrightarrow P_{k+1}$

Rules of the game

- How to measure error?

Rules of the game

- How to measure error?
- Energy norm: $|||v||| = |v|_{H_0^1(\Omega)}$

Rules of the game

- How to measure error?
- Energy norm: $|||v||| = |v|_{H_0^1(\Omega)}$
- How to measure complexity?

Rules of the game

- How to measure error?
- Energy norm: $|||v||| = |v|_{H_0^1(\Omega)}$
- How to measure complexity?
- Number of computations

Can we find "best" adaptive partitions

- Specify the methods more precisely

Can we find "best" adaptive partitions

- Specify the methods more precisely
- Admissible partitions: minimum angle condition and no hanging nodes

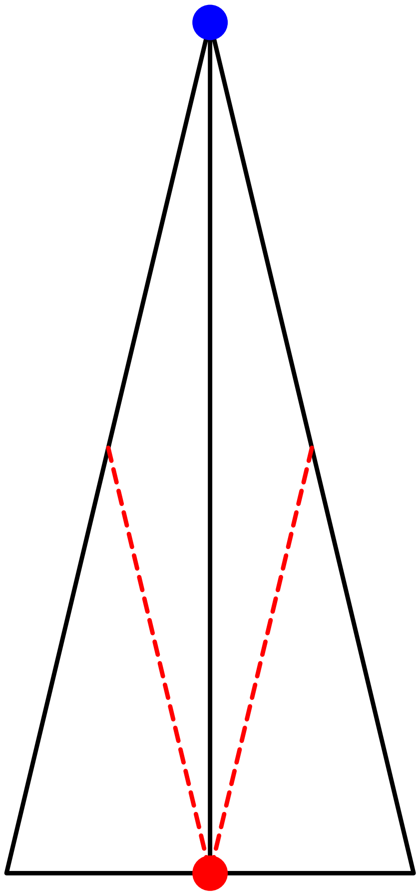
Can we find "best" adaptive partitions

- Specify the methods more precisely
- Admissible partitions: minimum angle condition and no hanging nodes
- Can almost do that

Can we find "best" adaptive partitions

- Specify the methods more precisely
- Admissible partitions: minimum angle condition and no hanging nodes
- Can almost do that
- Newest vertex bisection

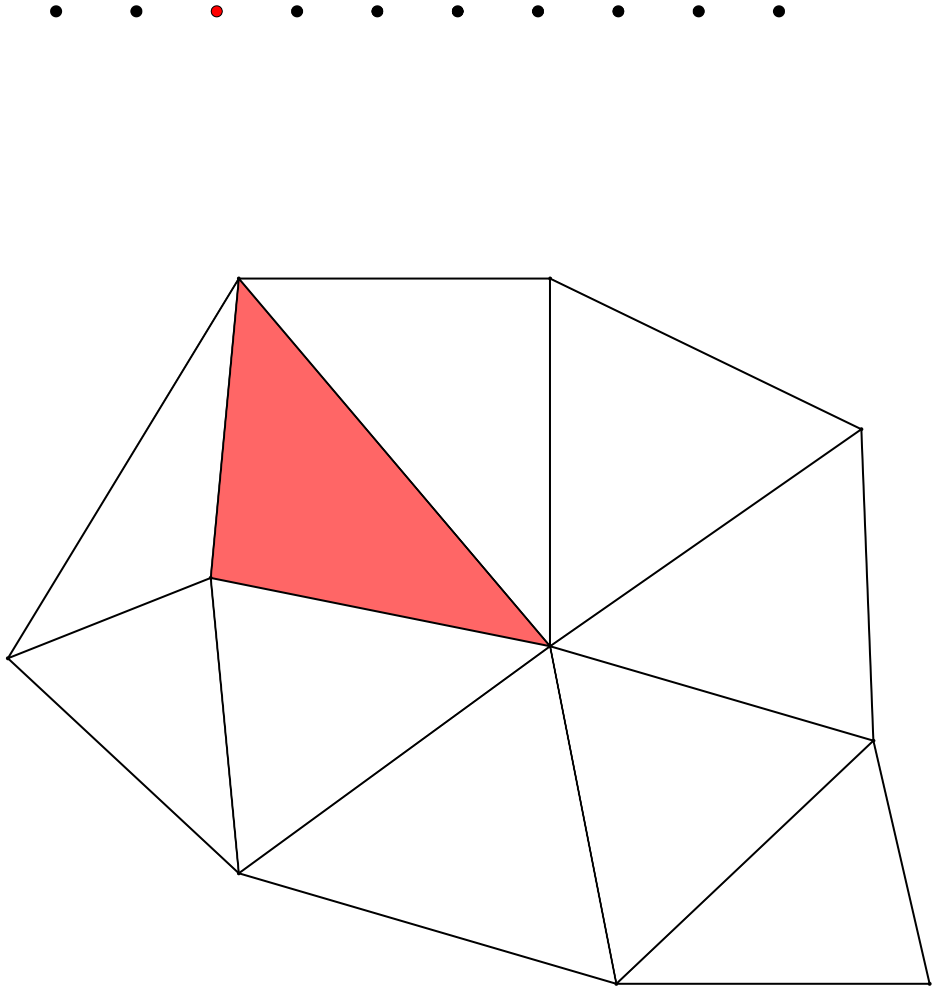
Newest vertex bisection



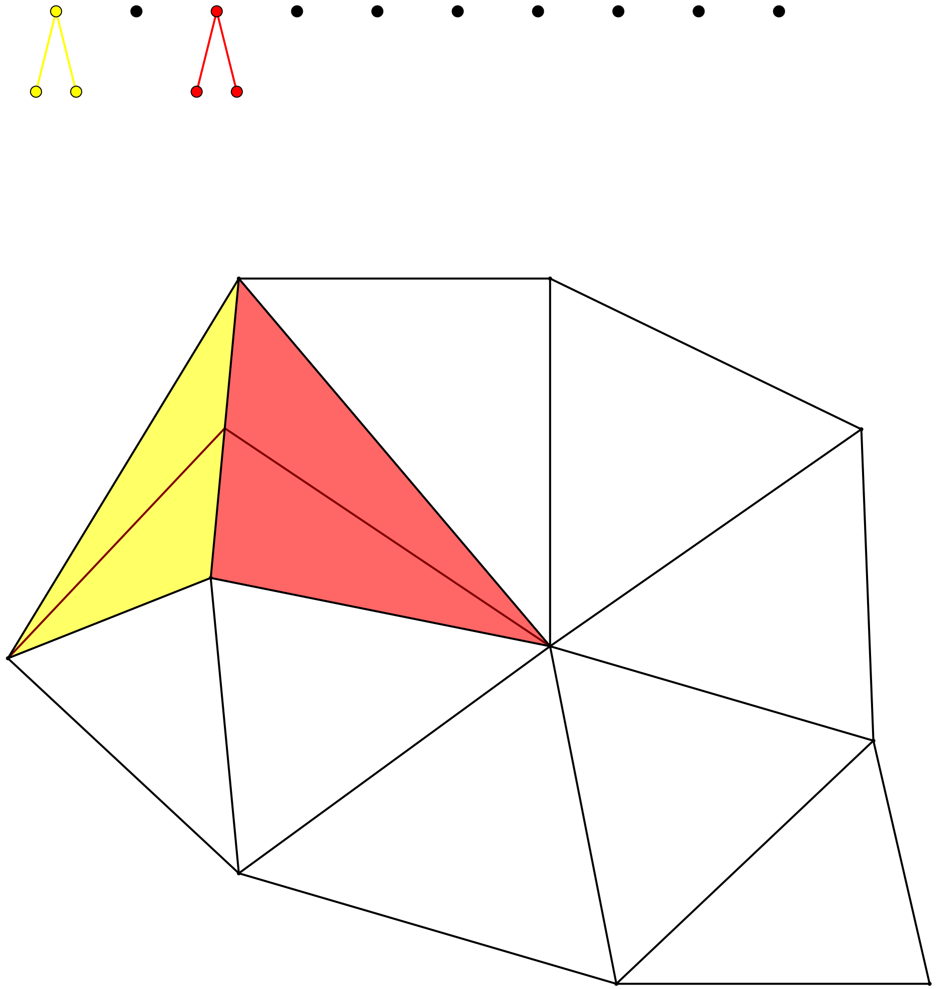
newest vertex of big triangle

newest vertex of new triangles

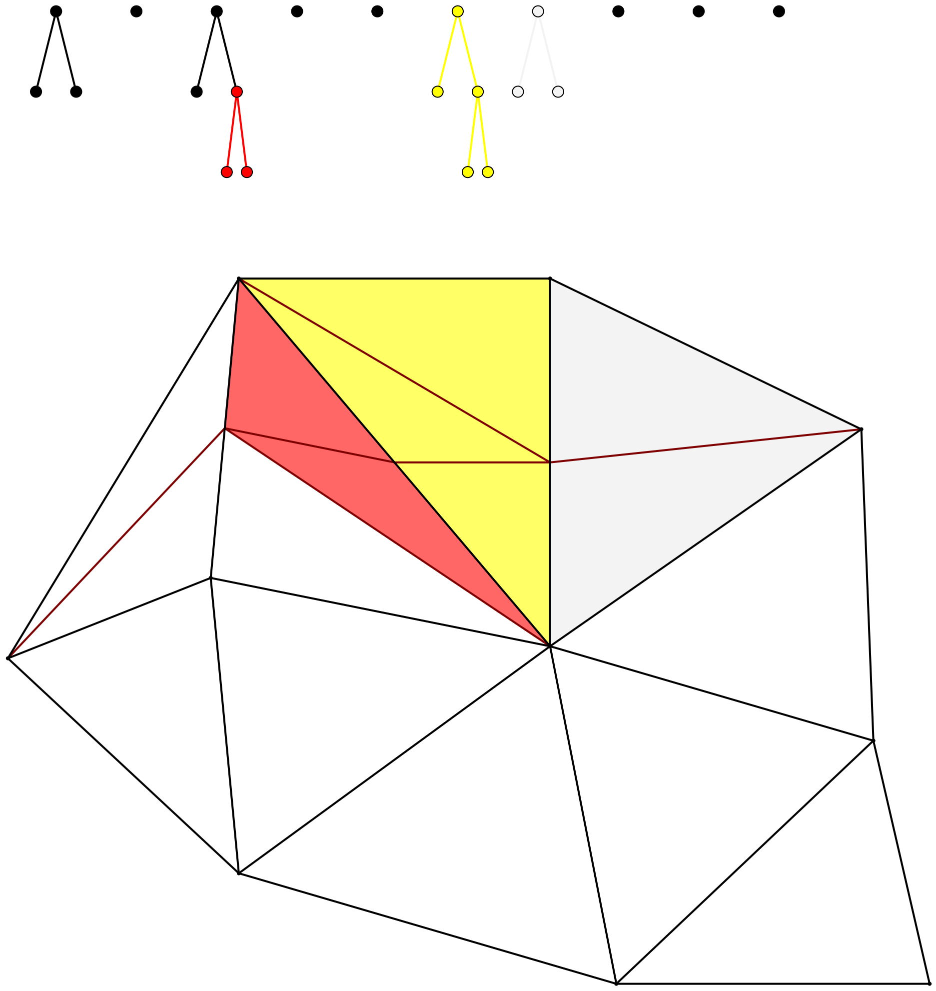
Tree structure



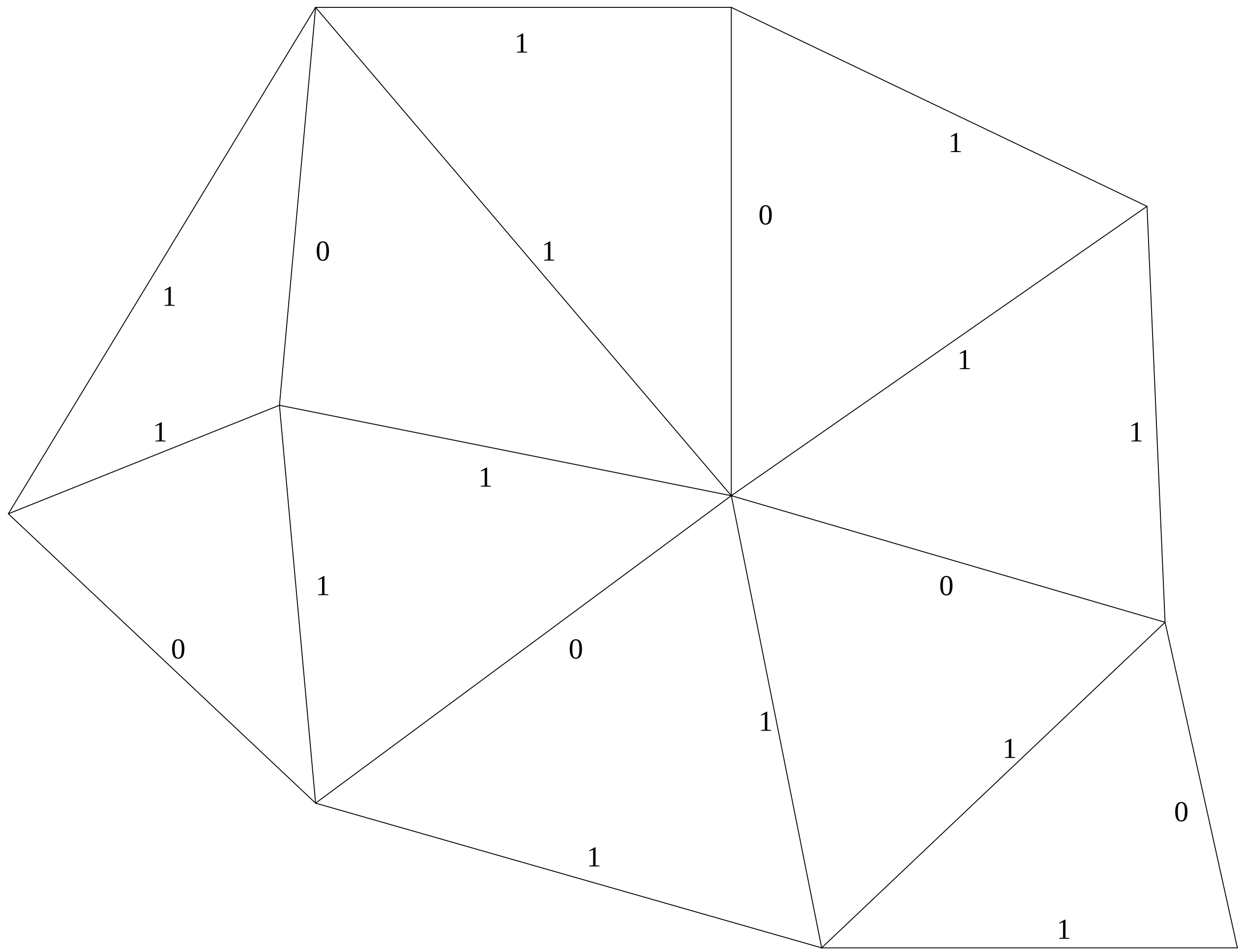
Refinement grows the tree



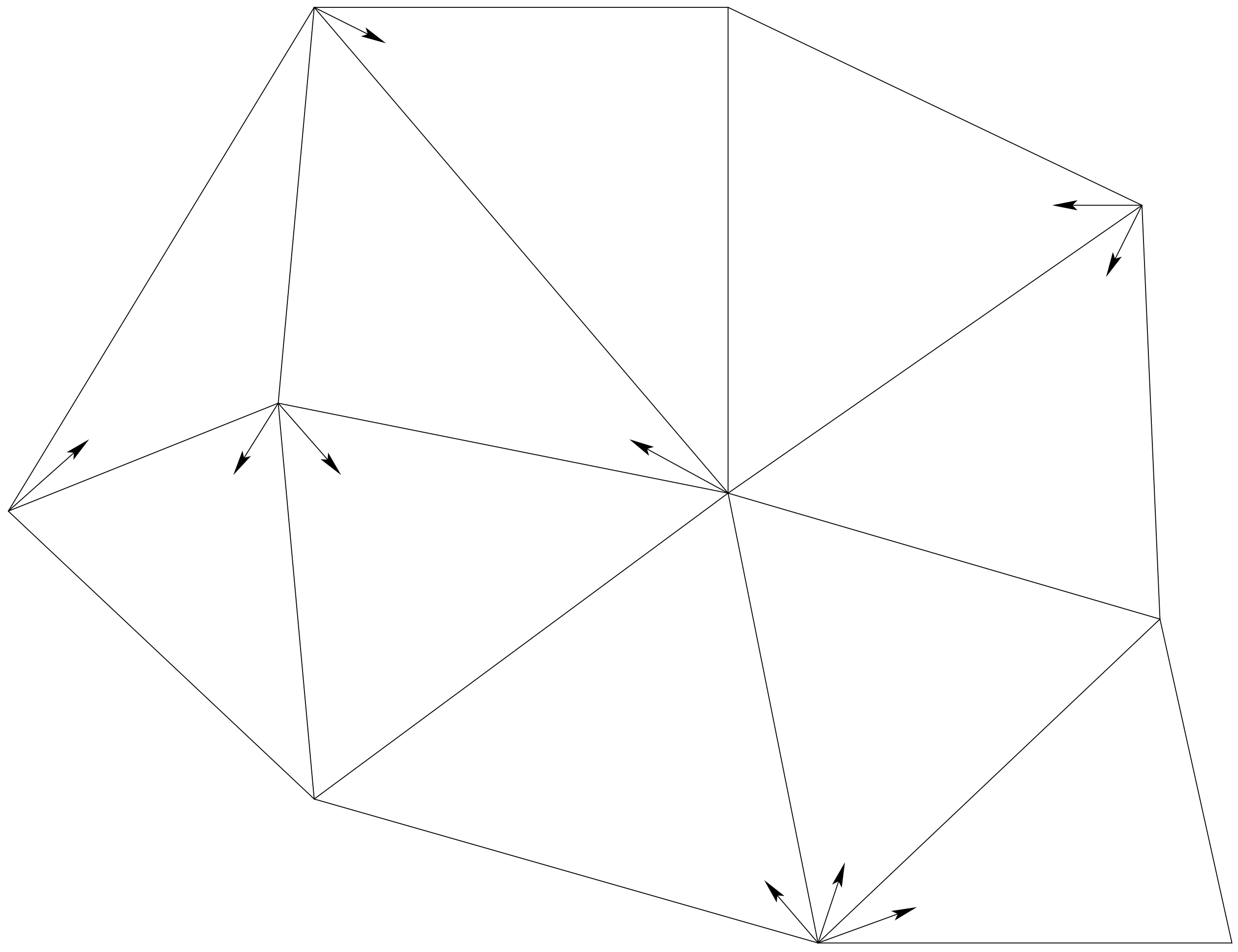
Grow more



Initial Labeling of edges



Initial assignment of newest vertices



Evaluate performance of an AFEM

- $N(P)$ number of subdivisions to create P

Evaluate performance of an AFEM

- $N(P)$ number of subdivisions to create P
- $\mathcal{P}_n := \{P : N(P) = n\}$

$$\sigma_n(u) := \inf_{P \in \mathcal{P}_n} \|u - u_P\|$$

Evaluate performance of an AFEM

- $N(P)$ number of subdivisions to create P
- $\mathcal{P}_n := \{P : N(P) = n\}$

$$\sigma_n(u) := \inf_{P \in \mathcal{P}_n} \|u - u_P\|$$

- Could ask Numerical Alg. perform like $\sigma_n(u)$

Evaluate performance of an AFEM

- $N(P)$ number of subdivisions to create P
- $\mathcal{P}_n := \{P : N(P) = n\}$

$$\sigma_n(u) := \inf_{P \in \mathcal{P}_n} \|u - u_P\|$$

- Could ask Numerical Alg. perform like $\sigma_n(u)$
- For $s > 0$, $\mathcal{A}^s$ is the set of all u such that

$$\sigma_n(u) \leq M n^{-s}, \quad n = 1, 2, \ldots$$

Evaluate performance of an AFEM

- $N(P)$ number of subdivisions to create P
- $\mathcal{P}_n := \{P : N(P) = n\}$

$$\sigma_n(u) := \inf_{P \in \mathcal{P}_n} \|u - u_P\|$$

- Could ask Numerical Alg. perform like $\sigma_n(u)$
- For $s > 0$, $\mathcal{A}^s$ is the set of all u such that

$$\sigma_n(u) \leq M n^{-s}, \quad n = 1, 2, \ldots$$

- Equivalent to $\sigma_n(u) \leq \epsilon$ with $n = C\epsilon^{-1/2}$

Binev-Dahmen- DeVore Algorithm

For each $\epsilon > 0$, algorithm produces P_ϵ such that

1. $|||u - u_{P_\epsilon}||| \leq \epsilon$

2. If $u \in \mathcal{A}^s$ then $\#(P_\epsilon) \leq C_0 |u|_{\mathcal{A}^s} \epsilon^{-1/s}$

3. Number of computations used is $\leq C_0 |u|_{\mathcal{A}^s} \epsilon^{-1/s}$

COROLLARY: The BDD AFEM beats Standard Finite Element Methods for a wide class of problems

How to do Marking

Bulk Chasing: Doerfler, Morin-Nochetto-Siebert

1. Uses error indicators $e(\Delta)$
2. Choose smallest set $\mathcal{M}_k$ such that

$$\sum_{\Delta \in \mathcal{M}_k} e(\Delta) \geq 1/2 \sum_{\Delta \in P_k} e(\Delta)$$

3. There exists $0 < \lambda < 1$: Given target accuarcy $\epsilon > 0$ and P_k which resolves f to accuracy $\gamma\epsilon$,. then either

$$|||u - u_{P_{k+1}}||| < \epsilon$$

or

$$|||u - u_{P_{k+1}}||| \leq \lambda |||u - u_{P_k}|||$$

Binev-Dahmen-DeVore Algorithm

$P_k \longrightarrow P_{k+1}$:

1. $P_{k,0} := P_k$

2. $P_{k,j-1} \longrightarrow P_{k,j}$, $j = 1, \ldots, K$:

$$|||u - u_{P_{k,j}}||| \leq \lambda |||u - u_{P_{k,j-1}}|||$$

3 $P_{k,K} \longrightarrow P_{k+1}$ by Coarsening

Two main results

- Refinements to remove hanging nodes do not inflate number of subdivisions severely

Two main results

- Refinements to remove hanging nodes do not inflate number of subdivisions severely
- How to do Coarsening

Control removal of hanging nodes

- $P_0 \rightarrow P_1 \rightarrow \cdots \rightarrow P_n$

Control removal of hanging nodes

- $P_0 \to P_1 \to \cdots \to P_n$
- $\mathcal{P}_k \to P_{k+1}$ uses $\mathcal{M}_k$ and $\mathcal{M}'_k$

Control removal of hanging nodes

- $P_0 \to P_1 \to \cdots \to P_n$
- $\mathcal{P}_k \to P_{k+1}$ uses $\mathcal{M}_k$ and $\mathcal{M}'_k$
- Theorem:

$$\#(P_n) \leq \#(P_0) + C(\#(\mathcal{M}_0) + \cdots + \#(\mathcal{M}_{n-1}))$$

Proof of Theorem

- Not simple induction: we do not have

$$\#\mathcal{M}'_k \leq C\#\mathcal{M}_k$$

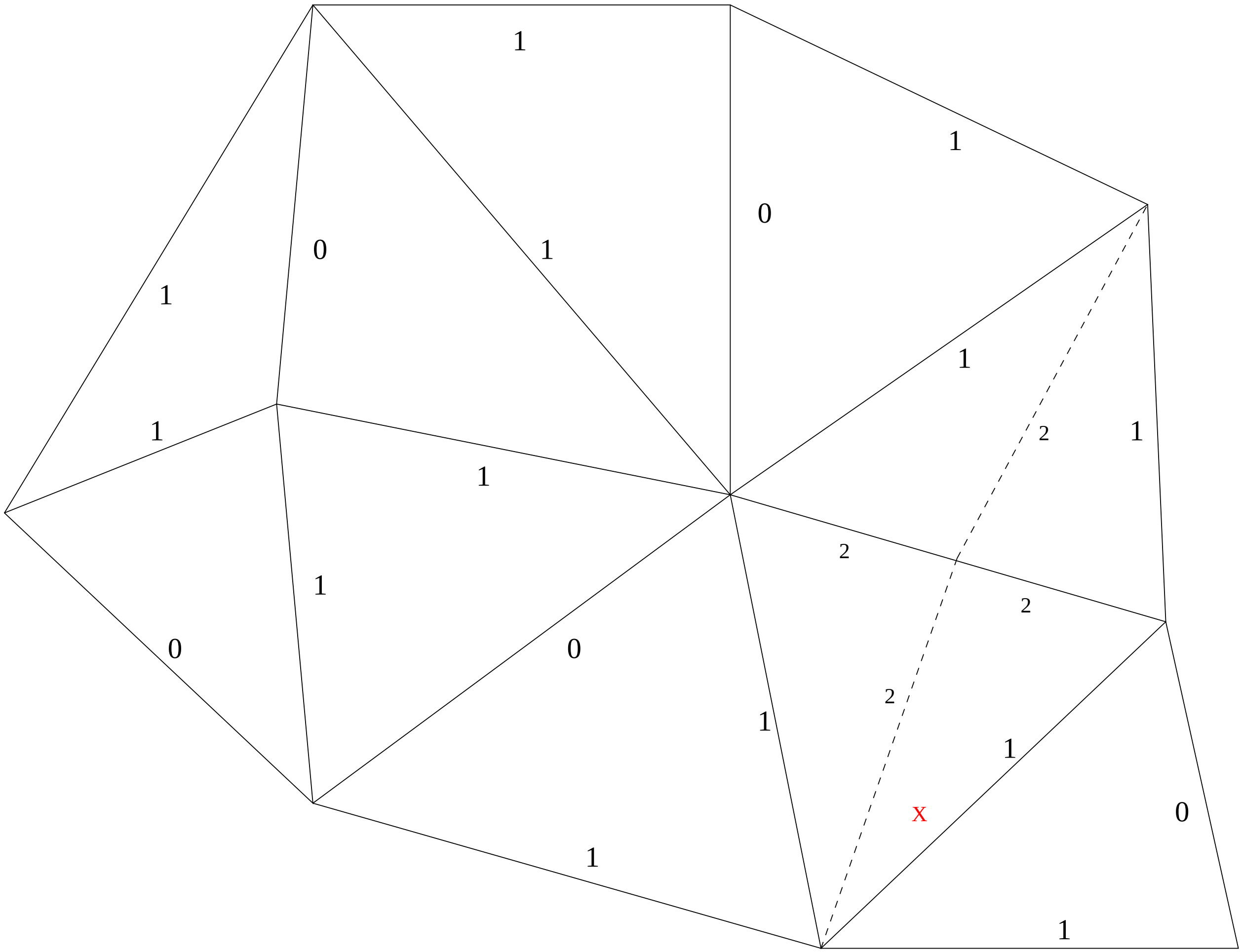
Proof of Theorem

- Not simple induction: we do not have

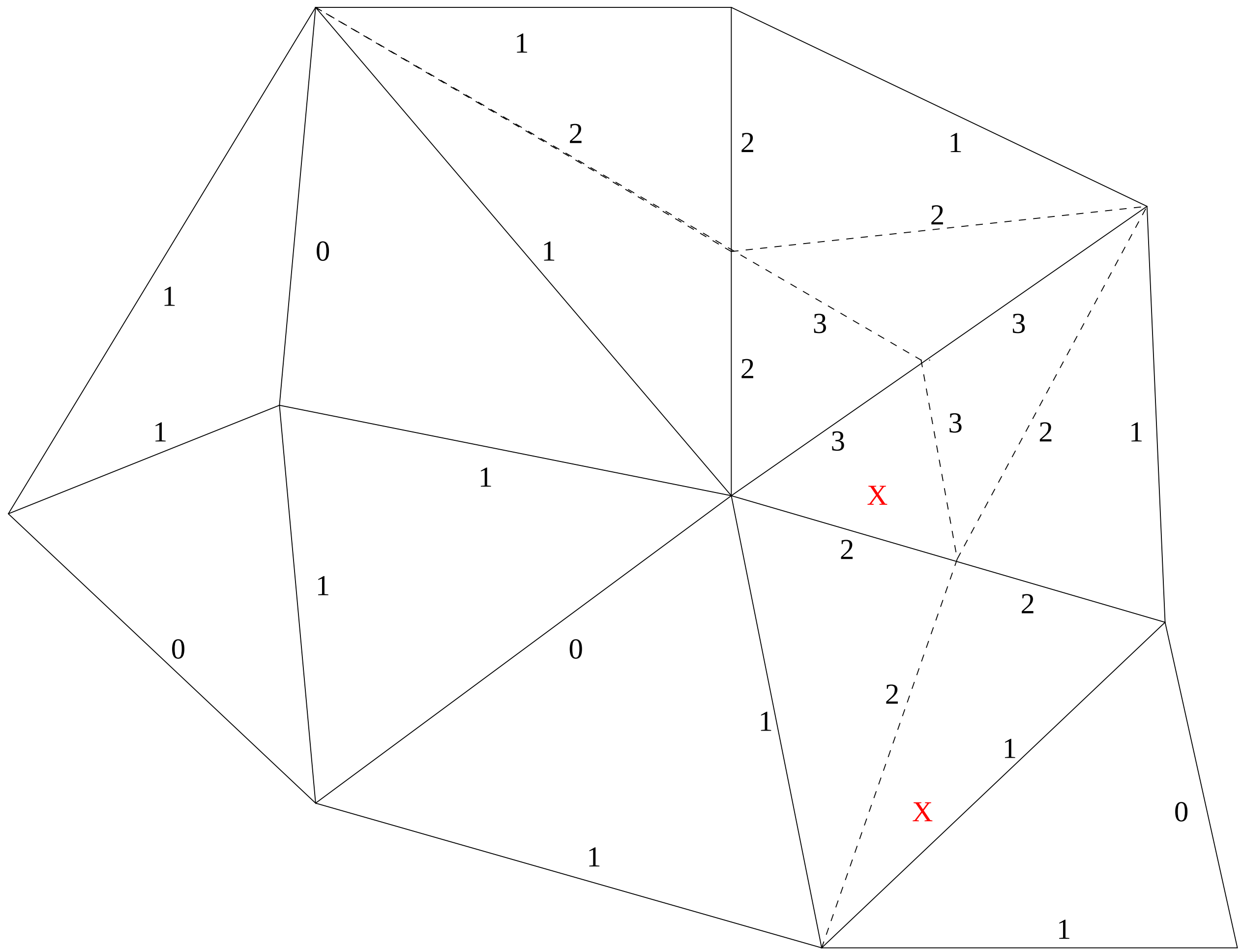
$$\#\mathcal{M}'_k \leq C\#\mathcal{M}_k$$

- Need to look at entire history of how $\Delta \in P_n$ was created

Many refinements



Many refinements

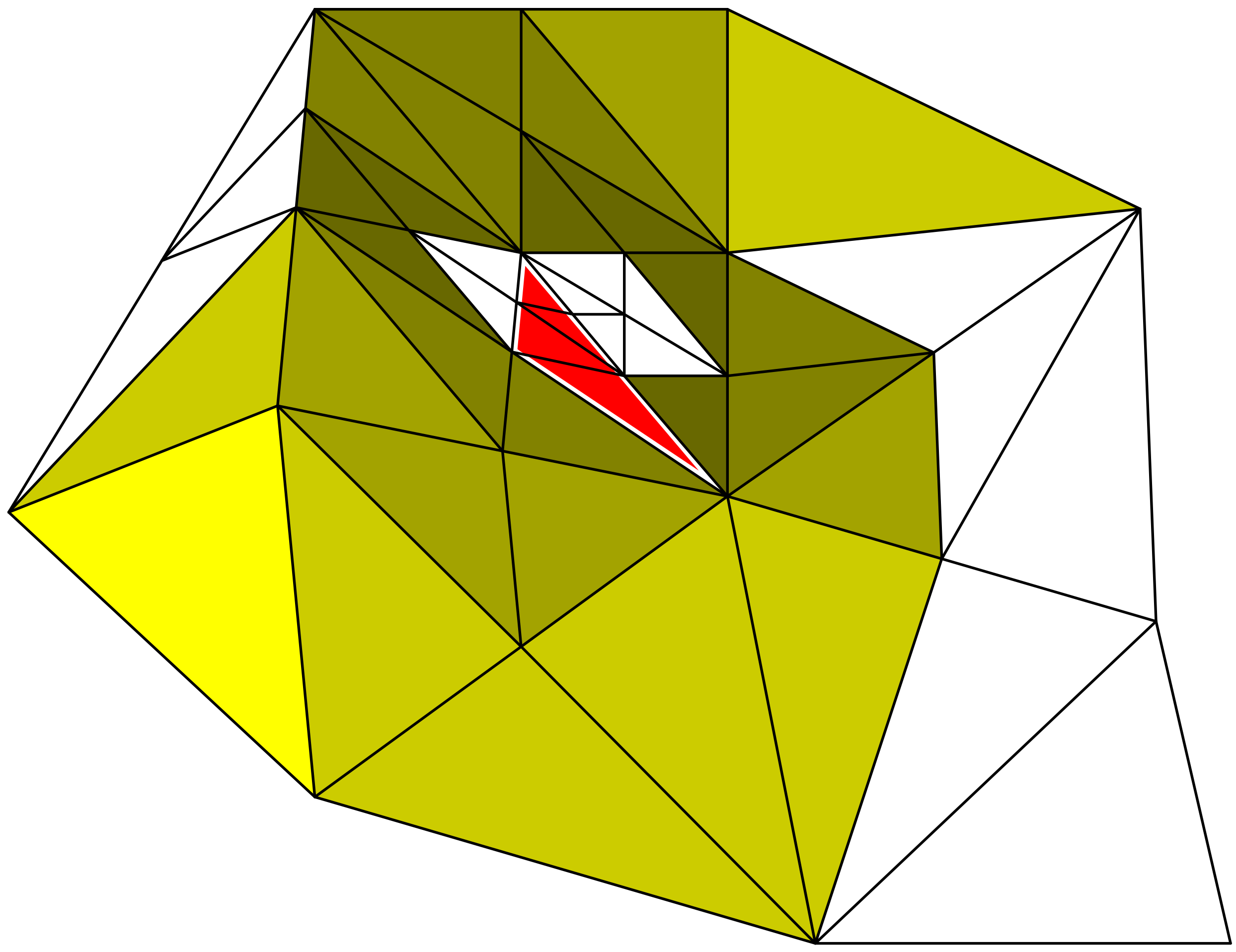


Ideas:

1. Each cell in $\mathcal{M} := \cup_{k=0}^{n-1} \mathcal{M}_k$ is given A dollars to spend
2. $\Delta' \in P_n$ receives $\lambda(\Delta, \Delta')$ dollars from a given $\Delta \in \mathcal{M}$.
Here λ depends on the generations $g(\Delta) = k$, $g(\Delta') = j$ and the distance $\mathrm{dist}(\Delta, \Delta')$

$$\lambda(\Delta, \Delta') := \begin{cases} (j-k+2)^{-2}, & \mathrm{dist} \leq A2^{-k/2}; k \leq j+1, \\ 0, & \text{otherwise}. \end{cases}$$

Who's got the money?



Ideas

3. Prove each $\Delta' \in P_n$ receives at least B dollars from all of the $\Delta \in \mathcal{M}$ combined.

4. The latter requires to look at how Δ' is created, i.e. the set of markings responsible for the creation of Δ'.

Coarsening Algorithm

- $P_{k,K}$ and $u_{P_{k,K}}$ known to us

Coarsening Algorithm

- $P_{k,K}$ and $u_{P_{k,K}}$ known to us
- $P_{k,K}$ may be too large

Coarsening Algorithm

- $P_{k,K}$ and $u_{P_{k,K}}$ known to us
- $P_{k,K}$ may be too large
- We want a sparse approximation to $u_{P_{k,K}}$ by using a smaller set than $P_{k,K}$

Coarsening Algorithm

- $P_{k,K}$ and $u_{P_{k,K}}$ known to us
- $P_{k,K}$ may be too large
- We want a sparse approximation to $u_{P_{k,K}}$ by using a smaller set than $P_{k,K}$
- Tree structure

Coarsening Algorithm

- $P_{k,K}$ and $u_{P_{k,K}}$ known to us
- $P_{k,K}$ may be too large
- We want a sparse approximation to $u_{P_{k,K}}$ by using a smaller set than $P_{k,K}$
- Tree structure
- master tree T_*: all cells Δ that can be obtained by newest vertex bisection

Tree approximation

- Error functional $e(\Delta) \geq 0$ defined on nodes Δ of master tree T_* -local error on cell

Tree approximation

- Error functional $e(\Delta) \geq 0$ defined on nodes Δ of master tree T_* -local error on cell
- Assumptions on e: if T is a tree with single root Δ

$$\sum_{\Delta' \in \mathcal{L}(T)} e(\Delta') \leq C\, e(\Delta)$$

Tree approximation

- Error functional $e(\Delta) \geq 0$ defined on nodes Δ of master tree T_* -local error on cell
- Assumptions on e: if T is a tree with single root Δ

$$\sum_{\Delta' \in \mathcal{L}(T)} e(\Delta') \leq C\, e(\Delta)$$

- Global error for tree T: $\mathcal{L}(T)$ leaves of T

$$E(T) := \sum_{\Delta \in \mathcal{L}(T)} e(\Delta)$$

Near best tree approximation

- Best approximation

$$\sigma_n := \inf_{N(T)=n} E(T)$$

Near best tree approximation

- Best approximation

$$\sigma_n := \inf_{N(T)=n} E(T)$$

- Near best tree T': $N(T') = n$

$$E(T') \leq C_1 \sigma_{c_1 n}$$

Near best tree approximation

- Best approximation

$$\sigma_n := \inf_{N(T)=n} E(T)$$

- Near best tree T': $N(T') = n$

$$E(T') \leq C_1 \sigma_{c_1 n}$$

- Theorem (Binev-DeVore): Given e, can find a near best tree with Cn computations

Ideas in proof

- Greedy strategy to subdivide those Δ with largest $e(\Delta)$ does not work

Ideas in proof

- Greedy strategy to subdivide those Δ with largest $e(\Delta)$ does not work
- After each subdivision, redefine e (modified $\tilde{e}$)

Ideas in proof

- Greedy strategy to subdivide those Δ with largest $e(\Delta)$ does not work
- After each subdivision, redefine e (modified $\tilde{e}$)
- Modification makes $\tilde{e}(\Delta)$ smaller when previous subdivisions of the ancestor's of Δ did not decrease error much

Ideas in proof

- Greedy strategy to subdivide those Δ with largest $e(\Delta)$ does not work
- After each subdivision, redefine e (modified $\tilde{e}$)
- Modification makes $\tilde{e}(\Delta)$ smaller when previous subdivisions of the ancestor's of Δ did not decrease error much
- Greedy on $\tilde{e}$